# 实验8&9、加法器、加减法器

# 和ALU基本原理与设计实验报告

姓名： 王祚滨 专业： 信息安全 学号： 3180104933

课程名称： 逻辑与计算机设计基础实验 同组学生姓名： 王国朝、赵卿云

指导老师： 洪奇军 实验地点： 浙江大学紫金港校区东四教学楼509实验日期：2019年 11 月 13 日

# 一、实验目的和要求

1.1掌握一位全加器的工作原理和逻辑功能

1.2掌握串行进位加法器的工作原理和进位延迟

1.3掌握减法器的实现原理

1.4掌握加减法器的设计方法

1.5掌握ALU基本原理及在CPU中的作用

1.6掌握ALU的设计方法

**二、实验内容和原理**

**2.1实验内容：**

任务1：原理图方式设计4位串行进位加法器

任务2：实现4位加法器应用

**2.2实验原理：**

**2.2.1位全加器**

三个输入位：数据位 Ai 和 Bi，低位进位输入 Ci

二个输出位：全加和 Si，进位输出 Ci+1

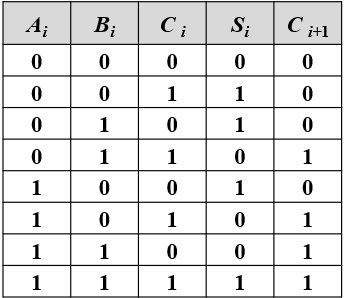




图1 一位全加器真值表和关系式

根据一位全加器的输入输出关系，得到电路图

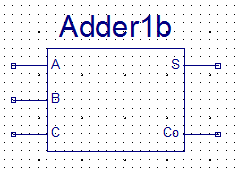
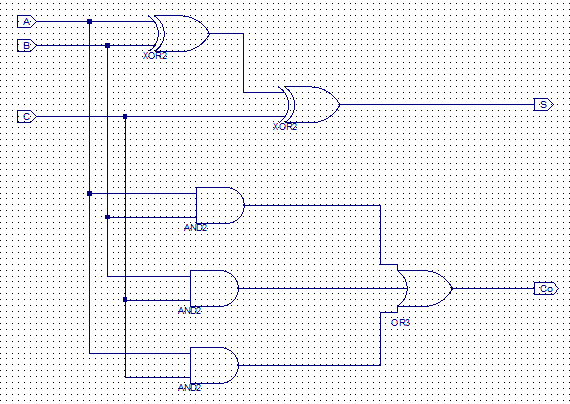
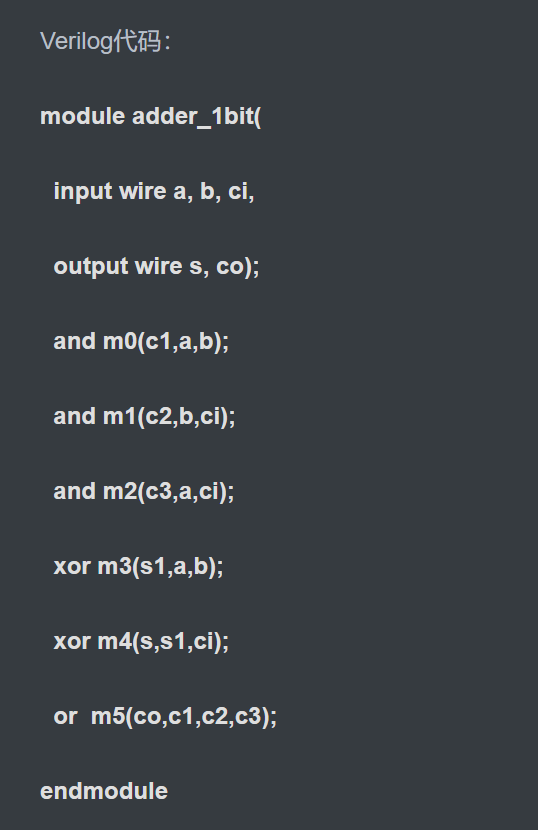


图2 一位加法器电路图



**2.2.2 多位串行进位加法器**

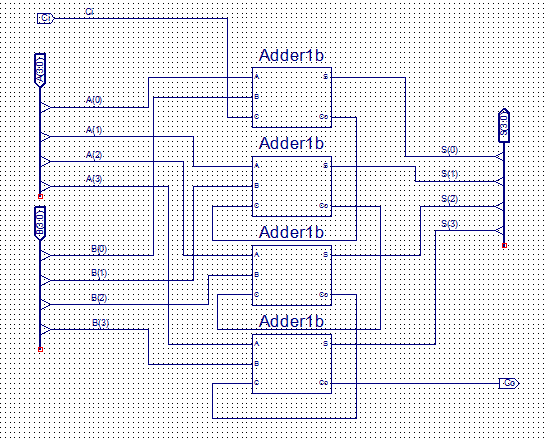
多位全加器可由一位全加器将进位串接构成

高位进位生成速度慢，位数越多时间越长



图3 多位全加器

**2.2.3 4位全加器**



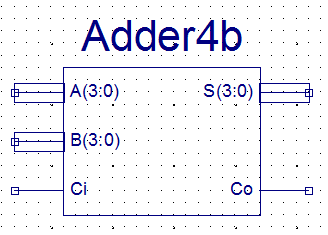


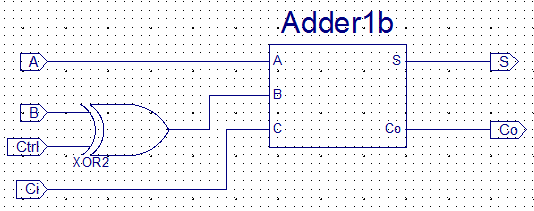
图4 四位全加器电路图

**2.2.4 1位加减法器**

用负数补码加法实现，减数当作负数求补码。

共用加法器。

用“异或”门控制求反，低位进位C0为1。



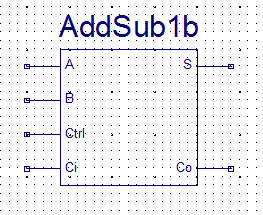


图5 一位加减法器

**2.2.5多位串行进位全减器**

用负数补码加法实现，减数当作负数求补码

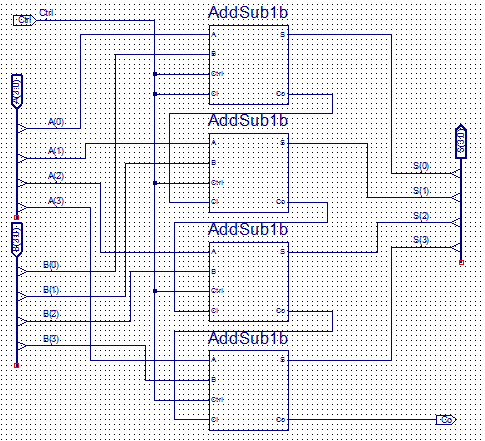
共用加法器

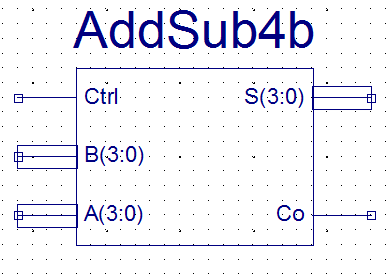
用“异或”门控制求反，低位进位C0为1





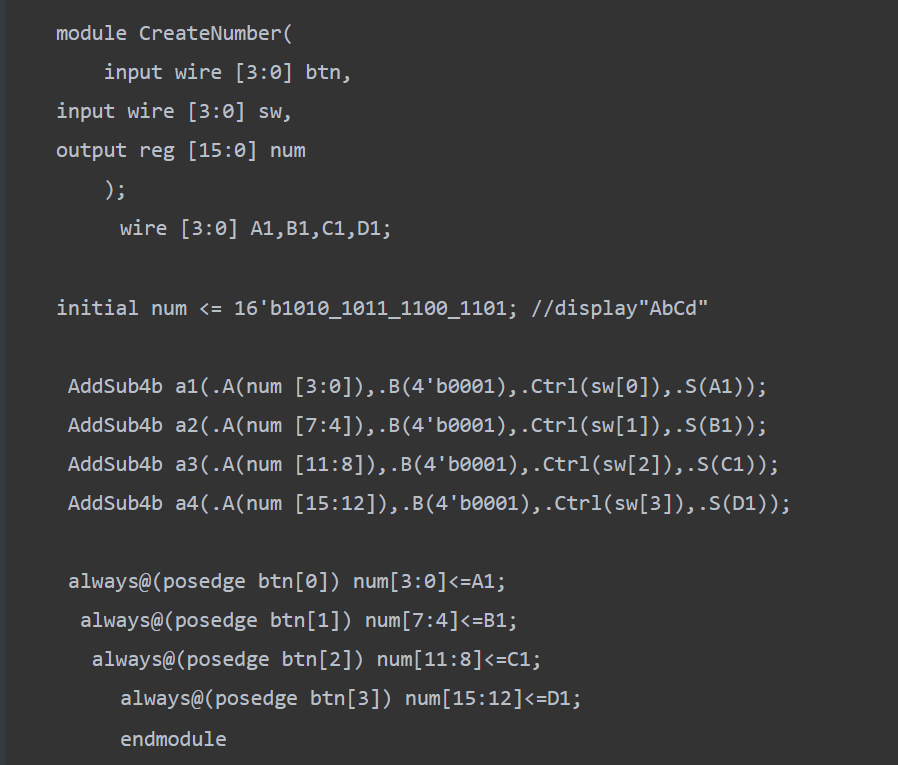
**2.2.6 4位加减法器**



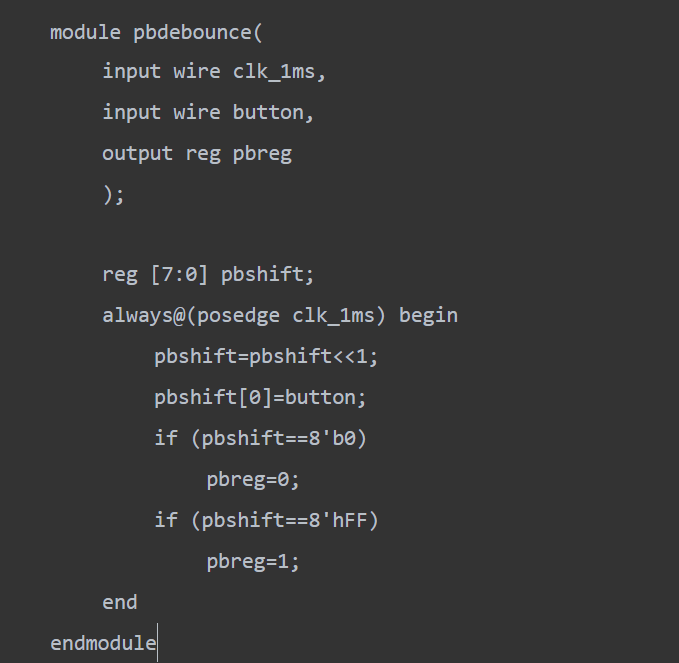


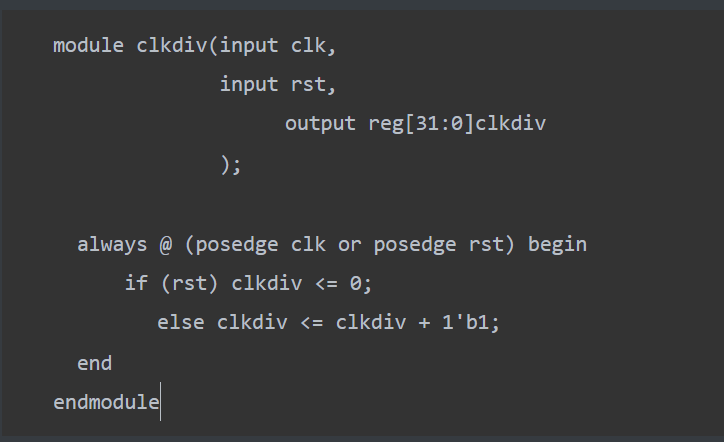
图六 四位加减法器

2.2.7 设计按键数据输入模块



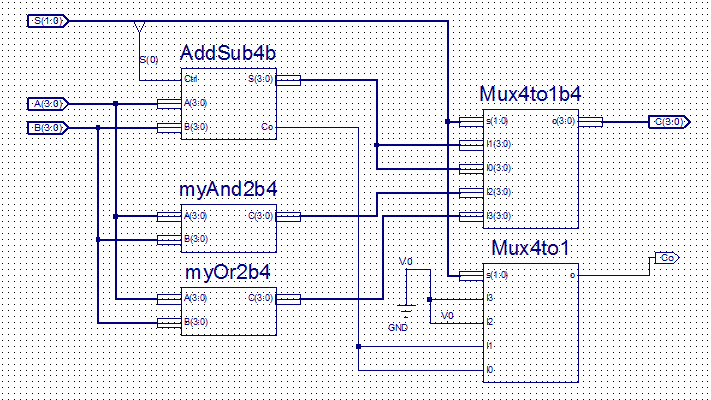
2.2.8 防抖动模块 + 分频器





图七 clkdiv图

**2.2.10 4位ALU原理图**



图八 2.2.10 4位ALU原理图

# 三、主要仪器设备

# 1. 装有Xilinx ISE 14.7的计算机 1台

# 2. SWORD开发板

# 四、操作方法与实验步骤

**4.1实验任务：**

任务1：原理图方式设计4位加减法器

任务2：实现4位ALU及应用设计

**4.2实验步骤：**

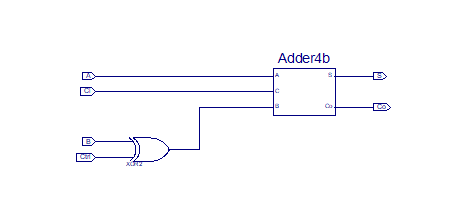
4.2.1 AddSub1b设计

1.新建工程，工程名称用MyALU。

2.Top Level Source Type用HDL

3.新建源文件，类型是Schematic，文件名称用AddSub1b。

4.原理图方式进行设计



图九 AdderSub1b

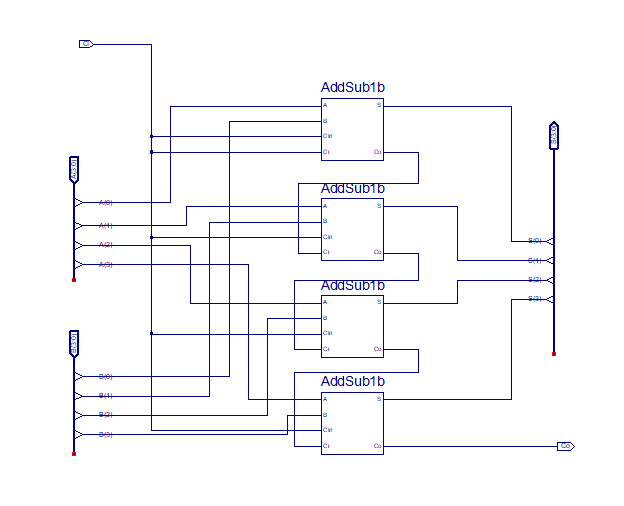
4.2.2 AddSub4b设计

1.新建源文件

2.类型是Schematic

3.文件名称用AddSub4b

4.原理图方式进行设计，调用前面设计的AddSub1b



图十 AdderSub4b

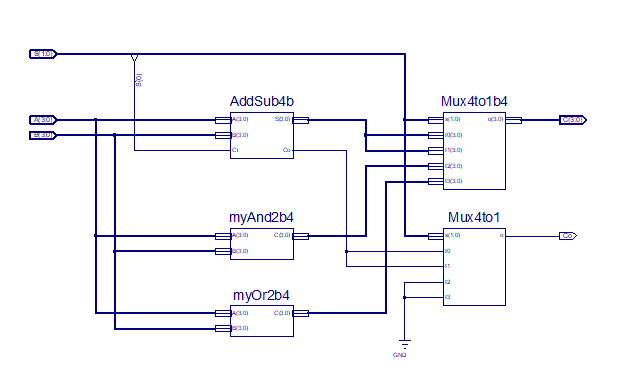
4.2.3 ALU设计

1.新建源文件

2.类型是Verilog或Schematic

3.文件名称用ALU

4.原理图方式进行设计



图十一 ALU

4.2.4 top设计

1.新建源文件，类型是Verilog，文件名Top。，右键设为“Set as Top Module”

2.代码输入进行设计

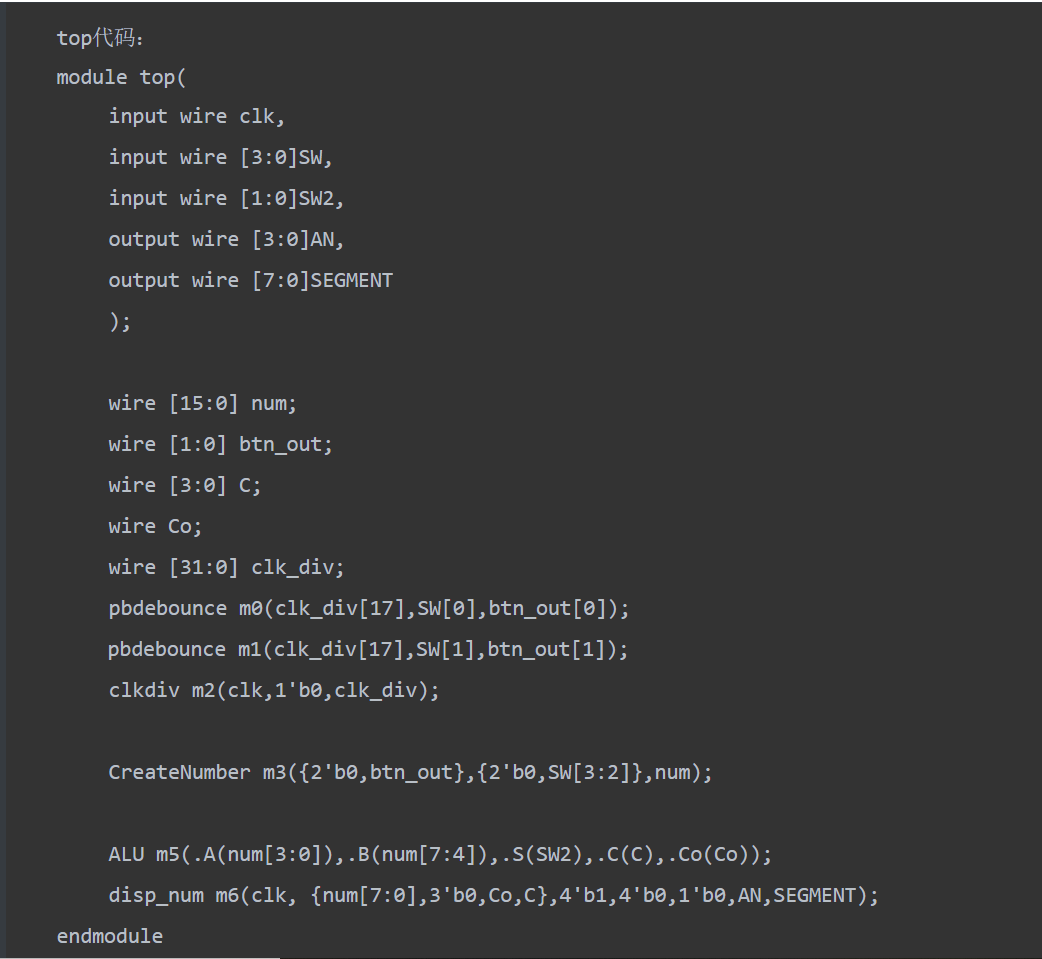
调用pbdebounce模块

调用AddSub4b模块

调用pbdebounce、clkdiv模块

调用DispNum模块

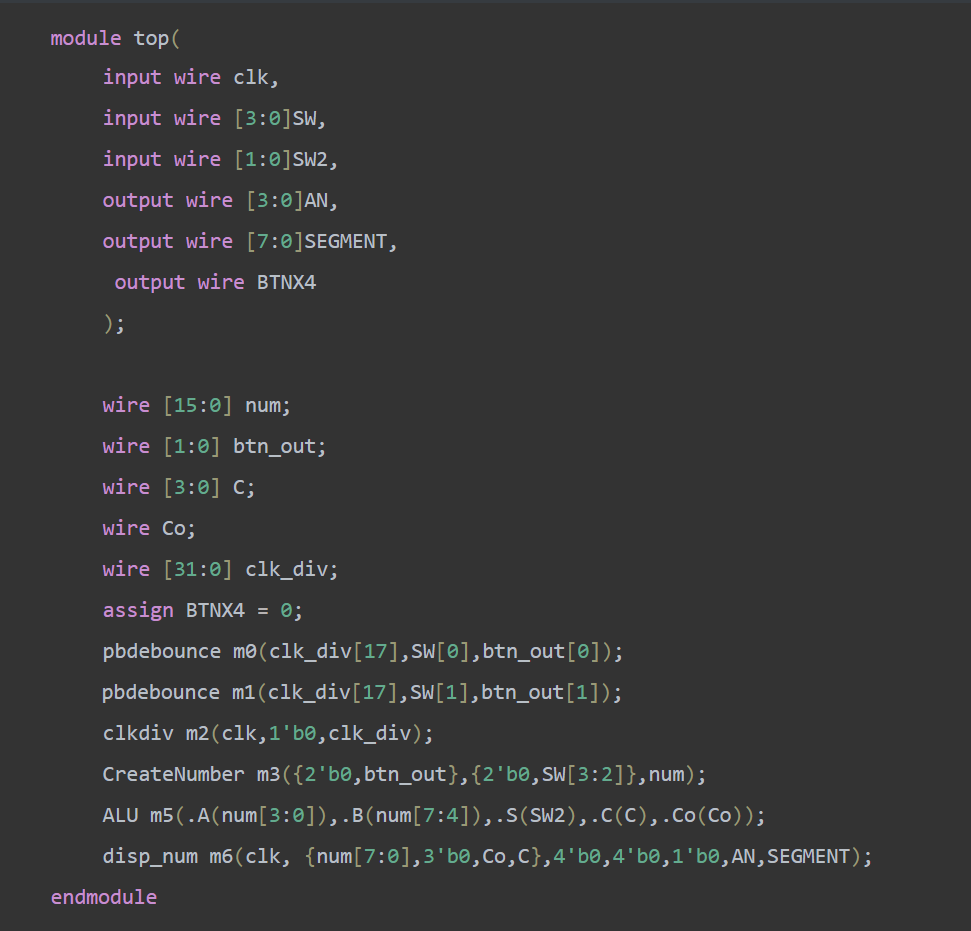
调用CreateNumber模块

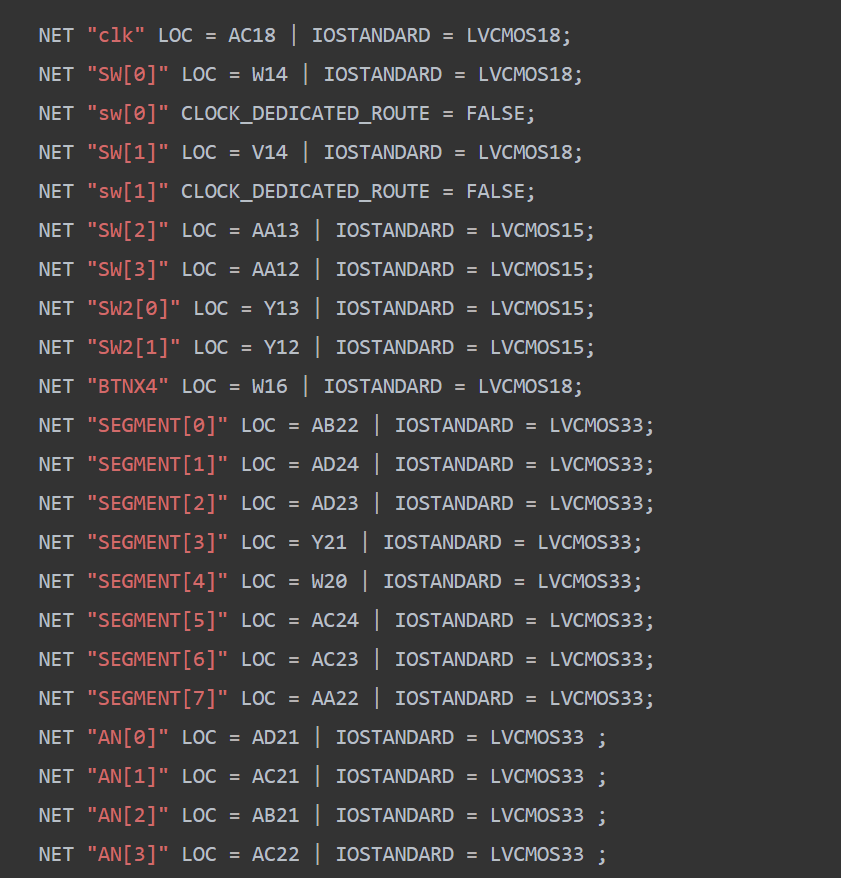


4.2.5. UCF引脚定义

# 

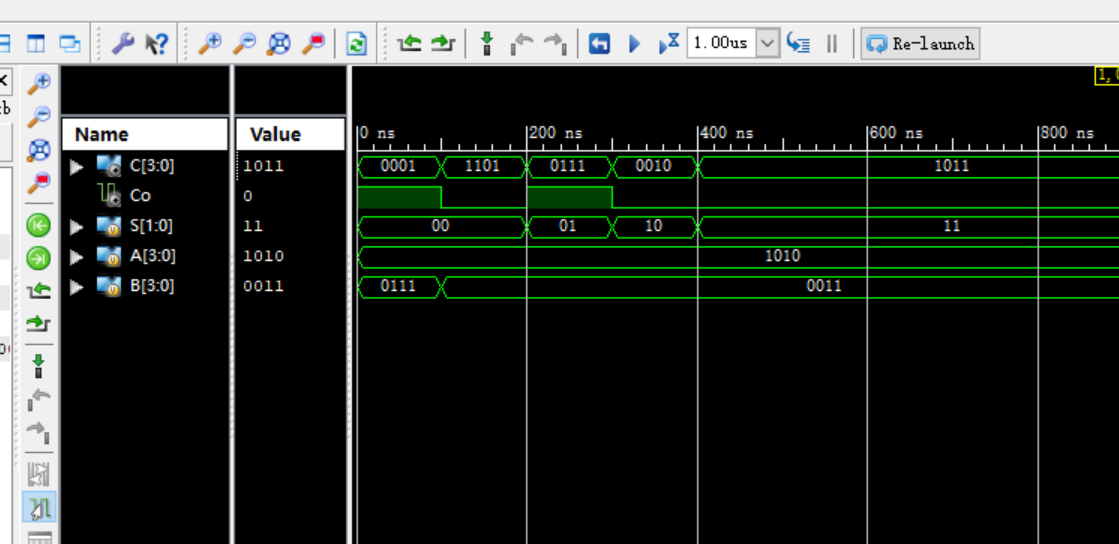
附加：更改为按钮实现





# 五、实验结果与分析

5.1 4bitALU仿真：



|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 时间 | 输入 | | | 计算 | | 输出 | |
| A[3:0] | B[3:0] | S[1:0] | 运算符 | 结果 | C0 | S[3:0] |
| 0-50ns | 1010 | 0111 | 00 | + | 1 0001 | 1 | 0001 |
| 50-100ns | 1010 | 0011 | 00 | + | 0 1101 | 0 | 1101 |
| 100-150ns | 1010 | 0011 | 01 | - | 1 0111 | 1 | 0111 |
| 150-200ns | 1010 | 0011 | 10 | AND | 0 0010 | 0 | 0010 |
| 200ns- | 1010 | 0011 | 11 | OR | 0 1011 | 0 | 1011 |

5.2 ALU计算结果

|  |  |  |
| --- | --- | --- |
| **数据** | **分析** | **结果** |
| IMG_20191113_200851 | **输入**  A[3:0] = 3 (0011)  B[3:0] = 6 (0110)  S[1:0] = 00  **计算**  Operator: +  Result = 9 (1001)  **输出**  C0 = 0  C[3:0] = 9 (1001) | **√** |
| IMG_20191113_200924 | **输入**  A[3:0] = F (1111)  B[3:0] = 6 (0110)  S[1:0] = 11  **计算**  Operator: OR  Result = F (0 1111)  **输出**  C0 = 0  C[3:0] = F (1111) | **√** |
| IMG_20191113_201007 | **输入**  A[3:0] = 2 (0010)  B[3:0] = 6 (0110)  S[1:0] = 01  **计算**  Operator: -  Result = -4  **输出**  C0 =1  C[3:0] = 4 | **√** |
| IMG_20191113_201012 | **输入**  A[3:0] =2 (0010)  B[3:0] = 6 (0110)  S[1:0] = 01  **计算**  Operator: AND  Result = 2 (0 0010)  **输出**  C0 = 0  C[3:0] = 2 (0010) | **√** |

# 六、讨论、心得

通过本次实验，我了解到了ALU的设计原理，同时对模块化的设计有了更深的了解，在实验中应用之前设计过的一些模块，让整个过程简单了很多。

在实现加减功能时，更是实践了课上所讲的补码形式，也对理论知识有了进一步的探究，总的来说实验难度并不大，只要细心就可以成功得出结果。

后期将开关改为按钮时，遇到了不小麻烦，没有将使能设置为0，可见对verilog语言仍然比较生疏，还需要进一步提高。